# object-hash

Generate hashes from objects and values in node and the browser. Uses node.js crypto module for hashing. Supports SHA1 and many others (depending on the platform) as well as custom streams (e.g. CRC32).

[NPM](https://www.npmjs.com/package/object-hash)

[Travis CI](https://secure.travis-ci.org/puleos/object-hash?branch=master) [Coverage Status](https://coveralls.io/github/puleos/object-hash?branch=master)

* Hash values of any type.
* Supports a keys only option for grouping similar objects with different values.

var hash = require('object-hash');

hash({foo: 'bar'}) // => '67b69634f9880a282c14a0f0cb7ba20cf5d677e9'

hash([1, 2, 2.718, 3.14159]) // => '136b9b88375971dff9f1af09d7356e3e04281951'

## Versioning Disclaimer

Starting with version 1.1.8 (released April 2017), new versions will consider the exact returned hash part of the API contract, i.e. changes that will affect hash values will be considered semver-major. Previous versions may violate that expectation.

For more information, see [this discussion](https://github.com/puleos/object-hash/issues/30).

## hash(value, options)

Generate a hash from any object or type. Defaults to sha1 with hex encoding.

* algorithm hash algo to be used: 'sha1', 'md5', 'passthrough'. default: sha1
  + This supports the algorithms returned by crypto.getHashes(). Note that the default of SHA-1 is not considered secure, and a stronger algorithm should be used if a cryptographical hash is desired.
  + This also supports the passthrough algorith, which will return the information that would otherwise have been hashed.
* excludeValues {true|false} hash object keys, values ignored. default: false
* encoding hash encoding, supports 'buffer', 'hex', 'binary', 'base64'. default: hex
* ignoreUnknown {true|\*false} ignore unknown object types. default: false
* replacer optional function that replaces values before hashing. default: accept all values
* respectFunctionProperties {true|false} Whether properties on functions are considered when hashing. default: true
* respectFunctionNames {true|false} consider name property of functions for hashing. default: true
* respectType {true|false} Whether special type attributes (.prototype, .\_\_proto\_\_, .constructor) are hashed. default: true
* unorderedArrays {true|false} Sort all arrays before hashing. Note that this affects *all* collections, i.e. including typed arrays, Sets, Maps, etc. default: false
* unorderedSets {true|false} Sort Set and Map instances before hashing, i.e. make hash(new Set([1, 2])) == hash(new Set([2, 1])) return true. default: true
* unorderedObjects {true|false} Sort objects before hashing, i.e. make hash({ x: 1, y: 2 }) === hash({ y: 2, x: 1 }). default: true
* excludeKeys optional function for excluding specific key(s) from hashing, if true is returned then exclude from hash. default: include all keys

## hash.sha1(value)

Hash using the sha1 algorithm.

Note that SHA-1 is not considered secure, and a stronger algorithm should be used if a cryptographical hash is desired.

*Sugar method, equivalent to* hash(value, {algorithm: 'sha1'})

## hash.keys(value)

Hash object keys using the sha1 algorithm, values ignored.

*Sugar method, equivalent to* hash(value, {excludeValues: true})

## hash.MD5(value)

Hash using the md5 algorithm.

Note that the MD5 algorithm is not considered secure, and a stronger algorithm should be used if a cryptographical hash is desired.

*Sugar method, equivalent to* hash(value, {algorithm: 'md5'})

## hash.keysMD5(value)

Hash object keys using the md5 algorithm, values ignored.

Note that the MD5 algorithm is not considered secure, and a stronger algorithm should be used if a cryptographical hash is desired.

*Sugar method, equivalent to* hash(value, {algorithm: 'md5', excludeValues: true})

## hash.writeToStream(value, [options,] stream)

Write the information that would otherwise have been hashed to a stream, e.g.:

hash.writeToStream({foo: 'bar', a: 42}, {respectType: false}, process.stdout)

// => e.g. 'object:a:number:42foo:string:bar'

## Installation

node:

npm install object-hash

browser: */dist/object\_hash.js*

<script src="object\_hash.js" type="text/javascript"></script>

<script>

var hash = objectHash.sha1({foo:'bar'});

console.log(hash); // e003c89cdf35cdf46d8239b4692436364b7259f9

</script>

## Example usage

var hash = require('object-hash');

var peter = { name: 'Peter', stapler: false, friends: ['Joanna', 'Michael', 'Samir'] };

var michael = { name: 'Michael', stapler: false, friends: ['Peter', 'Samir'] };

var bob = { name: 'Bob', stapler: true, friends: [] };

/\*\*\*

\* sha1 hex encoding (default)

\*/

hash(peter);

// 14fa461bf4b98155e82adc86532938553b4d33a9

hash(michael);

// 4b2b30e27699979ce46714253bc2213010db039c

hash(bob);

// 38d96106bc8ef3d8bd369b99bb6972702c9826d5

/\*\*\*

\* hash object keys, values ignored

\*/

hash(peter, { excludeValues: true });

// 48f370a772c7496f6c9d2e6d92e920c87dd00a5c

hash(michael, { excludeValues: true });

// 48f370a772c7496f6c9d2e6d92e920c87dd00a5c

hash.keys(bob);

// 48f370a772c7496f6c9d2e6d92e920c87dd00a5c

/\*\*\*

\* hash object, ignore specific key(s)

\*/

hash(peter, { excludeKeys: function(key) {

if ( key === 'friends') {

return true;

}

return false;

}

});

// 66b7d7e64871aa9fda1bdc8e88a28df797648d80

/\*\*\*

\* md5 base64 encoding

\*/

hash(peter, { algorithm: 'md5', encoding: 'base64' });

// 6rkWaaDiG3NynWw4svGH7g==

hash(michael, { algorithm: 'md5', encoding: 'base64' });

// djXaWpuWVJeOF8Sb6SFFNg==

hash(bob, { algorithm: 'md5', encoding: 'base64' });

// lFzkw/IJ8/12jZI0rQeS3w==

## Legacy Browser Support

IE <= 8 and Opera <= 11 support dropped in version 0.3.0. If you require legacy browser support you must either use an ES5 shim or use version 0.2.5 of this module.

## Development

git clone https://github.com/puleos/object-hash

## Node Docker Wrapper

If you want to stand this up in a docker container, you should take at look at the [node-object-hash](https://github.com/bean5/node-object-hash) project.

### gulp tasks

* gulp watch (default) watch files, test and lint on change/add
* gulp test unit tests
* gulp karma browser unit tests
* gulp lint jshint
* gulp dist create browser version in /dist

## License

MIT

## Changelog

### v2.0.0

Only Node.js versions >= 6.0.0 are being tested in CI now. No other breaking changes were introduced.